Oikos

o19831

Hu, Y. H., Sha, L. Q., Blanchet, F. G., Zhang, J. L., Tang, Y., Lin, Y. C. Lan, G. Y. and Cao, M. 2012. Dominant species and dispersal limitation regulate tree species distributions in a 20-ha plot in Xishuangbanna, southwest China. – Oikos 121: 952–960.

Appendix 1

Table A1. Taxonomic distributions of tree species in five DBH classes.

	Total no. of modelled species	Total no. of genera	Total no. of families
Class 0 (DBH ≥ 1 cm)	191	122	53
Class 1 (1 cm ≤ DBH < 5 cm)	147	103	45
Class 2 (5 cm ≤ DBH < 10 cm)	61	51	29
Class 3 (10 cm ≤ DBH < 25 cm)	57	48	29
Class 4 (DBH ≥ 25 cm)	22	19	16

Appendix 2

This program is used for conducting model selection by simulated annealing method.

```
SA.r=function(y,X,nmutation=200,nstart=5,Temp=.75,inter.0=TRUE,
print.counter=TRUE,plot.res=TRUE){
###
### This function use the simulated annealing, with the AIC, to find the best
### variables to include in the SAR model.
###
### The simulated annealing is an optimization approach developped independently
### by Kirkpatrick et al. (1983) and Cerny (1985).
###
### Arguments:
###
### y = Numeric vector. A response variable.
### X = Matrix or data.frame. All the explanatory variables that maybe included
###in the model.
### inter.0 = Logical. Whether the intercept of the model should be force to 0 or not.
### nmutation = Number of mutations to be performed. Default is 200. See Details
###for explanation of the mutation implemented.
### nstart = Number of new model construction to be performed. Default is 5
### Temp = Number defining if a model, which is not the best, should be kept.
###This number will need to be fixed for every new model in order to
###obtaine a good model fairly fast (See Details). Default is 1.
### print.counter = Logical. Whether a counter presenting the number of mutation
```

```
###carried out so far should be printed in on the screen.
###Default is TRUE.
### plot.res = Logical. If the result should be plotted or not. Useful to define
###nmutation and Temp arguments. Default is TRUE.
###
### Details:
###
### In this function, three different mutations were implemented: "add",
### "remove", and "change".
###
### The "add" mutation allows a new variable to enter the model, if there are
### any variable not already entered. If there is no new variable, one variable
### is randomly removed, to mutate the model. The new variable to enter is also
### chosen randomly.
###
### The "remove" mutation take out a new variable from the model. If there is
### only one variable in the model, either "add" or "change" is selected,
### randomly, to mutate the model. The removed variable is also chosen randomly.
###
### The "change" mutation allows a variable already present in the model to be
### replaced by another one not present in the model. The variable to be change
### is selected randomly. The replacing variable is also selected randomly. If
### there is no new variable, one variable is randomly removed, to mutate the
### model.
###
```

```
### In this R function, the algorithm will always select a new model if it is
### better (that is, if it has a lower AIC than the previously best one).
### However, if the model is not as good as the previous one, a probability
### function is calculated to evaluate if the model should be kept or not or
### not. In this implementation, the probability of keeping a new model that is
### worst than the best one so far is :
###
### exp((AIC.best-AIC.new)/(Temp/((number of mutation carried out so far)/nmutation)))
###
### This probability function was proposed by Krikpatrick et al. (1983). To make
### sure that a minimum number of mutation is needed in the
### optimization, it is necessary to make a few tries with different Temp, thus
### deciding which will give the best ratio time/precision.
###
### Copyleft - F. Guillaume Blanchet - October 2009 - version 0.4
##############################################################################
###


    "%w/o%" <- function(x,y) x[!x %in% y] #--    x without y


    if(!is.numeric(y)){
        stop("'y' is not numeric")
    }


    if(!all(apply(X,2,is.numeric))){
```

```
        stop("One or more variables in 'X' is not numeric")
    }


    ### Result object
    res<-list()

    ### Possible modifcation done to the model
    modif<-c("add","remove","change")

    for(l in 1:nstart){
        ### Initiate object
        k<-1
        aic.best<-numeric()
        Xform.best<-character()


        ### Initial model (step)
        if(inter.0){
            Xform<-c(-1,colnames(X)[1])
            Xform.best[k]<-paste(Xform,collapse="+")
            form<-as.formula(paste(c("y~",Xform.best)))
        }
        if(!inter.0){
            Xform<-colnames(X)[1]
```

```
            Xform.best[k]<-Xform
            form<-as.formula(paste("y~",Xform.best))
        }

        ### calculating AIC (Energy)

    sar1<-glm(form,family=Gamma(link='inverse'),data=X,start=rep(.5,len=c(length(Xform))+1))
        aic.best[k]<- sar1$aic

        for(i in 1:nmutation){

#print(Xform)
            ### Perform a mutation
            mutate<-sample(modif,1)
            varia<-unlist(strsplit(as.character(Xform),split="\\+"))
            ### Add a new variable
            if(mutate=="add"){
                if(length(varia)==ncol(X) & !inter.0){
                    ### Remove an old variable
                    Xnew<-sample(varia,length(varia)-1)
                    Xform<-paste(Xnew,collapse="+")
                }
                if(length(varia)==(ncol(X)+1) & inter.0){
                    ### Remove an old variable, and keep "-1"
```

```
            Xnew<-""
            while(!any(Xnew=="-1")){
                Xnew<-sample(varia,length(varia)-1)
            }
            Xform<-paste(Xnew,collapse="+")
        }
        if((length(varia)<ncol(X) & !inter.0) | (length(varia)<(ncol(X)+1) & inter.0)){
            ### Possible choice
            Xchoice<-colnames(X)%w/o%varia

            ### Add a new variable
            Xnew<-sample(Xchoice,1)
            Xform<-paste(c(Xform,Xnew),collapse="+")
        }
    }

    ### Remove an old variable
    if(mutate=="remove"){
        if(length(varia)==1 & !inter.0){
            mutate<-sample(modif[c(1,3)],1)

            if(mutate=="add"){
                ### Possible choice
                Xchoice<-colnames(X)%w/o%varia
```

```
                        ### Add a new variable
                        Xnew<-sample(Xchoice,1)
                        Xform<-paste(c(Xform,Xnew),collapse="+")
                }
                if(mutate=="change"){
                        ### Possible choice
                        Xchoice<-colnames(X)%w/o%varia

                        ### Change a variable
                        Xnew<-sample(Xchoice,1)
                        pos<-sample(1:length(varia),1)

                        varia[pos]<-Xnew
                        Xform<-paste(varia,collapse="+")
                }
        }
        if(length(varia)==2 & inter.0){
                mutate<-sample(modif[c(1,3)],1)

                if(mutate=="add"){
                        ### Possible choice
                        Xchoice<-colnames(X)%w/o%varia

                        ### Add a new variable
                        Xnew<-sample(Xchoice,1)
```

```
                        Xform<-paste(c(Xform,Xnew),collapse="+")
                    }
                    if(mutate=="change"){
                        ### Possible choice
                        Xchoice<-colnames(X)%w/o%varia

                        ### Change a variable
                        Xnew<-sample(Xchoice,1)
                        poss.pos<-(1:length(varia))[-which(varia=="-1")]

                        if(length(poss.pos)==1){
                            pos<-poss.pos
                        }
                        if(length(poss.pos)>1){
                            pos<-sample(poss.pos,1)
                        }

                        varia[pos]<-Xnew
                        Xform<-paste(varia,collapse="+")
                    }
                }
                if(length(varia)>1 & !inter.0){
                    ### Remove an old variable
                    Xnew<-sample(varia,length(varia)-1)
                    Xform<-paste(Xnew,collapse="+")
```

```
            }
            if(length(varia)>2 & inter.0){
                ### Remove an old variable, and keep "-1"
                Xnew<-""
                while(!any(Xnew=="-1")){
                    Xnew<-sample(varia,length(varia)-1)
                }
                Xform<-paste(Xnew,collapse="+")
            }
        }

        ### Change an old variable for a new one
        if(mutate=="change"){
            if(length(varia)==ncol(X) & !inter.0){
                ### Remove an old variable
                Xnew<-sample(varia,length(varia)-1)
                Xform<-paste(Xnew,collapse="+")
            }
            if(length(varia)==(ncol(X)+1) & inter.0){
                ### Remove an old variable, and keep "-1"
                Xnew<-""
                while(!any(Xnew=="-1")){
                    Xnew<-sample(varia,length(varia)-1)
                }
                Xform<-paste(Xnew,collapse="+")
```

```
        }


        if(length(varia)<ncol(X) & !inter.0){
            ### Possible choice
            Xchoice<-colnames(X)%w/o%varia


            ### Change a variable
            Xnew<-sample(Xchoice,1)
            pos<-sample(1:length(varia),1)


            varia[pos]<-Xnew
            Xform<-paste(varia,collapse="+")
        }
        if(length(varia)<(ncol(X)+1) & inter.0){
            ### Possible choice
            Xchoice<-colnames(X)%w/o%varia


            ### Change a variable
            Xnew<-sample(Xchoice,1)
            poss.pos<-(1:length(varia))[-which(varia=="-1")]


            if(length(poss.pos)==1){
                pos<-poss.pos
            }
            if(length(poss.pos)>1){
```

```
                        pos<-sample(poss.pos,1)
                    }

                    varia[pos]<-Xnew
                    Xform<-paste(varia,collapse="+")
                }
            }

            form<-as.formula(paste("y~",Xform))

            ### calculating AIC (Energy)

    sar1<-glm(form,family=Gamma(link='inverse'),data=X,start=rep(.5,len=c(length(strsplit(a
s.character(Xform),split="\\+")[[1]]))+1))
            aic<- sar1$aic

            if(aic.best[k] >= aic){
                form.best<-form
                k<-k+1
                aic.best[k]<-aic
                Xform.best[k]<-Xform
            }

            if(aic.best[k] < aic){
                prob<-exp((aic.best[k]-aic)/(Temp/(i/nmutation)))
```

```
                if(prob > runif(1)){
                    form.best<-form
                    k<-k+1
                    aic.best[k]<-aic
                    Xform.best[k]<-Xform
                }
            }
            ### Print a counter
            if(print.counter & ((nmutation/i)==2 | (nmutation/i)==4 | (nmutation/i)==6 | (nmutation/i)==8)){
                print(i)
            }
        }

        if(plot.res){
            if(length(grep("darwin",R.Version()$os))!=1){
                X11()
            }else{
                quartz()
            }
            plot(aic.best,type="b")
        }

        ### Final object
        res[[l]]<-data.frame(Xform.best,aic.best)
```

```
        colnames(res[[l]])<-c("model","AIC")
    }
    names(res)<-paste("start",1:nstart,sep=".")


    return(res)
}
```

References:

Kirkpatrick, S. et al. 1983. Opimization by simulated annealing. – Science 220: 671-680.

Cerny, V. 1985. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. – J. Optimiz. Theory App. 45: 41-51.

Appendix 3

Characteristics of the 20 species with the highest importance values.

Rank	Species	Relative density	Relative frequency	Relative dominance	Importance value	Total basal area (m^2)
1	*Pittosporopsis kerrii*	21.81	1.98	3.59	27.38	30.72
2	*Parashorea chinensis*	8.27	1.75	13.27	23.29	113.54
3	*Garcinia cowa*	4.52	1.93	2.28	8.73	19.49
4	*Mezzettiopsis creaghii*	3.44	1.21	3.00	7.65	25.62
5	*Castanopsis echidnocarpa*	0.92	0.43	5.71	7.06	48.88
6	*Baccaurea ramiflora*	3.35	1.79	1.70	6.84	14.58
7	*Knema furfuracea*	3.30	1.78	1.32	6.40	11.32
8	*Sloanea tomentosa*	0.53	0.94	4.84	6.30	41.41
9	*Saprosma ternata*	2.82	1.65	0.13	4.60	1.11
10	*Phoebe lanceolata*	2.51	1.39	0.63	4.52	5.35
11	*Pometia tomentosa*	0.50	0.99	2.76	4.24	23.59
12	*Nephelium chryseum*	1.15	1.41	1.52	4.09	13.04

Rank	Species	Relative density	Relative frequency	Relative dominance	Importance value	Total basal area (m^2)
13	*Cinnamomum bejolghota*	1.39	1.62	1.03	4.04	8.78
14	*Diospyros hasseltii*	0.85	1.17	1.46	3.48	12.51
15	*Ficus langkokensis*	1.40	1.06	0.89	3.35	7.65
16	*Dichapetalum gelonioides*	1.28	1.40	0.30	2.99	2.60
17	*Castanopsis hystrix*	0.25	0.49	2.22	2.97	19.00
18	*Castanopsis megaphylla*	0.27	0.56	2.14	2.97	18.27
19	*Semecarpus reticulata*	0.65	1.07	1.22	2.93	10.41
20	*Chisocheton siamensis*	0.76	1.28	0.63	2.67	5.39

Appendix 4

p-values from comparisons of the explained variances among the five DBH classes.

	Class 0	Class 1	Class 2	Class 3	Class 4
Class 0	-	-	-	-	-
Class 1	0.0009***	-	-	-	-
Class 2	0.1949	0.2062	-	-	-
Class 3	0.6070	0.01423*	0.2436	-	-
Class 4	0.1824	0.0008***	0.0197*	0.0678	-

Note: p-values are derived from Kruskal–Wallis rank-sum tests on the variances explained by the most parsimonious models for each species. Class 0 (DBH ≥ 1 cm), Class 1 (1 cm ≤ DBH < 5 cm), Class 2 (5 cm ≤ DBH < 10 cm), Class 3 (10 cm ≤ DBH < 25 cm), Class 4 (DBH ≥ 25 cm).

Appendix 5

Relationships between species distributions and neighbourhood index for trees in each of the five DBH classes.

DBH class	Total no. of modelled species	Total no. of species with distributions related to neighbourhood index	Total no. of positive relationships	Total no. of negative relationships	Total no. of significant relationships
Class 0 (DBH ≥ 1 cm)	191	104 (54.4%)	53	51	54 (28.3%)
Class 1 (1 cm ≤ DBH < 5 cm)	147	46 (31.3%)	30	16	20 (13.6%)
Class 2 (5 cm ≤ DBH < 10 cm)	61	20 (32.8%)	11	9	8 (13.1%)
Class 3 (10 cm ≤ DBH < 25 cm)	57	25 (43.9%)	12	13	14 (26.4%)
Class 4 (DBH ≥ 25 cm)	22	15 (68.2%)	7	8	6 (27.3%)

Appendix 6

Relationships between species distributions and the three groups of explanatory variables for trees in each of the five DBH classes.

DBH class	Explanatory variable group	No. of positive coefficients	No. of negative coefficients
Class 0 (DBH ≥ 1 cm)	Group 1	1600	1419
	Group 2	716	513
	Group 3	736	838
Class 1 (1 cm ≤ DBH < 5 cm)	Group 1	789	795
	Group 2	415	195
	Group 3	500	477
Class 2 (5 cm ≤ DBH < 10 cm)	Group 1	472	428
	Group 2	186	103
	Group 3	204	261
Class 3 (10 cm ≤ DBH < 25 cm)	Group 1	414	402
	Group 2	115	124
	Group 3	186	236
Class 4 (DBH ≥ 25 cm)	Group 1	280	189
	Group 2	111	78
	Group 3	122	143

Note: Group 1 represents the soil explanatory variables; Group 2 represents the explanatory variables of topography and gap; Group 3 represents the explanatory variables of dominant neighbours and neighbourhood index.